DOCUMENT RESUME

ED 108 597 IR 002 108

AUTHOR	Porch, Ann
TITLE	Language Analysis Package (L.A.P.) Version I System Design.
INSTITUTION	Southwest Regional Laboratory for Educational Research and Development, Los Alamitos, Calif.
REPORT NO	SWRL-TM-5-72-06
PUB DATE	20 Apr 72
NOTE	52p.
EDRS PRICE	MF-$0.76 HC-$3.32 PLUS POSTAGE
DESCRIPTORS	Computer Programs; *Computer Science; Electronic Data Processing; Program Design; *Programing Languages; *Program Planning; Research Needs; Systems Analysts; *Systems Concepts
IDENTIFIERS	*Language Analysis Package; LAP

ABSTRACT

To permit researchers to use the speed and versatility of the computer to process natural language text as well as numerical data without undergoing special training in programing or computer operations, a language analysis package has been developed partially based on several existing programs. An overview of the design is provided and system functions, data-management functions, special parameter functions, and data-processing functions described in detail. Transaction language control also is explained. A bibliography is appended. (SK)

SWRL

SOUTHWEST REGIONAL LABORATORY TECHNICAL MEMORANDUM

DATE: April 20, 1972

NO: TM 5-72-06

TITLE: LANGUAGE ANALYSIS PACKAGE (L.A.P.) VERSION I SYSTEM DESIGN

AUTHOR: Ann Porch

ABSTRACT

An overview of design algorithms and architecture of a package of computer programs to handle natural language analysis is presented, with each subsystem of the package described in detail.

DOCUMENTATION OUTLINE

LANGUAGE ANALYSIS PACKAGE (L.A.P.) VERSION I SYSTEM DESIGN

1.0.0 INTRODUCTORY CONCEPTS

For years there have been "packaged programs" in statistical areas. These programs offer generalized computational capabilities in a form and format especially suited to easy use by researchers whose basic orientation is not that of Computer Science.[1] Researchers in the social sciences, for instance, are able to perform complex multi-variate regression analysis by computer without undergoing any special training in programming or computer operations.

A Language Analysis Package (L.A.P.) with a power comparable to that of statistical packages will have considerable general utility.

It will permit researchers to use the speed and versatility of the computer to process natural language text as well as numerical data. For example, researchers doing studies of textbooks typically analyze their data by hand. Where computer technology is employed, a special-purpose program is generally written by the resident programmer, who may not have specialized training in techniques of natural language processing. The results are costly, both in time and money spent on processing with inefficient or one-shot programs. Because such programs are limited in scope and written for a special purpose, the researcher finds that a relatively minor change in his research perspective makes the computer program unusable.

---

[1] See Dixon, W. J. (ed.), Biomedical Computer Programs, Univ. of Calif. Press, (1970) and Nie, N. et al. (ed.), Statistical Package for the Social Sciences, McGraw Hill, (1970).

During the past ten years, a great deal of work has been done in natural language processing throughout the world in fields such as artificial intelligence, information retrieval, machine translation, computational linguistics, and computer stylistics. Hundreds of computer programs have been written, debugged, run, and then shelved when the researcher went on to another project. A number of these programs are the product of months of careful work by experts.

1.1.0 Rationale for Package Development

The design proposed here suggests the use of the best of those existing programs whose authors are willing to release them, together with programs written specially for the package. Such an approach has several advantages: the package will reflect the power of the finest specialized programming skill presently available; the development costs will be minimized, since one major programming task will consist of interfacing the existing programs or subsections in a modular fashion under the direction of one control routine, rather than developing each of the specialized routines from scratch. By carefully constructing the package in a highly independent, modular manner, individual routines may be easily "un-plugged" and replaced when a more efficient or powerful routine is developed or should a new approach indicate combinations of functions. Thus, the system will be dynamic and open-ended, capable of being easily updated to keep pace with the state-of-the-art.

The package will be developed in several stages. This document primarily describes the limited capability of Version I, but will often refer to more powerful capabilities to be incorporated in later versions.

Care will be taken to "tag" these future features and to differentiate them from the system design for Version I.

All development will try to keep the entire package as machine independent as possible. Certainly it will be implemented on a computer which can compile and run most of the major computer languages currently in use for language processing. One such installation exists at UCLA, where the IBM 360 mod 91 has compilers for the following languages: PL/1, FORTRAN IV (G & H), COBOL, SNOBOL, LISP, APL, 360 ASSEMBLER, and ALGOL. Its operating system also allows for interfacing subroutines written in different source languages.

Certain hardware and system software requirements will be essential to the efficient development of the L.A.P. Among these are an efficient system sort-merge routine, multiple tape drives, relatively large amounts of direct-access storage and considerable available core. Again, the UCLA installation is one example of a computer center which is amply equipped in all areas.

Subsequent versions of the Language Analysis Package will have the ability to perform efficiently in all basic areas of natural language analysis, and the flexibility to operate either on-line or in batch mode. In addition, beginning with Version I, the L.A.P. will be easily modifiable at any time, either for more effective general use or for a particular research application.

Flexibility of use will be a major consideration in the development of such a package. Version I of the L.A.P. will provide the user with many options, allowing him to select precisely and easily only the functions he requires. No section of the package will be

"called-in" unless the researcher specifically requests it. He will not be limited to simply an exclusive "OR" type selection, where he can only chose to do either a KWIC or an Index, but will be able to combine routines and subroutines in the logical order he desires. He may, for example, want to produce KWIC's on words occurring within his inclusion list while simultaneously producing an index of all words except those in his exclusion list and a frequency count of every word in the text. He will be able to use only the retrieval aspect of the L.A.P. or only the statistical portion without being penalized by the fact he is using a package rather than a single program designed specifically for his purpose.

1.2.0 Types of Processing for Natural Language

A survey of the work currently being done in the field of language analysis[2] reveals seven major areas of present interest and usage which can logically be included in Version I of the L.A.P. Of the seventy-five projects listed in the November, 1971 issue of Computers and the Humanities, nine dealt with frequency counts, seventeen with KWIC production, fourteen with semantic or content analysis (including automatic abstracting), eight with statistics, thirteen with index production, six with retrieval systems, six with sentence parsing and six with miscellaneous items such as machine translation. Several projects must be considered to fall into more than one category. Ratios in Linguistics in Documentation (Current

[2] See Porch, Ann, "People and Projects in Natural Language Processing: A Preliminary Bibliographic Directory" TM 5-71-10, August 5, 1971, 107 pages.

Abstracts), Language and Automation, and Computer Studies in the Humanities and Verbal Behavior are much the same, although with a slightly heavier emphasis on retrieval and parsing.

1.3.0 Data Bases and Their Manipulation

Since a number of researchers will be using the L.A.P., Version I of the system will have the ability to differentiate among data bases, selecting the researcher's base or sub-base from a library of resident data bases stored on magnetic tape or disc and making it available for his processing. In later versions, the entire library may include the ERIC files, or state adopted textbooks. Researchers using the L.A.P. will be able to obtain an input file containing only first grade reading books or only ERIC documents dealing with reading.

Often, data bases that a particular researcher may wish to use have been prepared elsewhere with each having different input conventions and formats. For example, one data base might have been prepared with a logical record length of 100 and in EBCDIC code, utilizing both upper and lower case characters, while another might have been prepared with a logical record length of 72, in ASCII code, and be in upper case only with capitals indicated by a "/" preceding the capitalized letter. Version I of the L.A.P. will be able to handle input formats and sets of conventions that the researcher can specify (See 3.5.0 Translate Module).

A researcher may want to use output from one step in the modular L.A.P. execution as input to another. He may want to select subsets of a given data base, process each separately, then cross reference the results or subsets of the results. He may want to do transformations on the data

as it is being processed, and use the transformed data as input. Version I of the L.A.P. will be able to save output for further processing and will be able to save subsets of data once they are selected, in order to save the expense of repeated retrieval processing. The researcher will be able to present the system with a new file, or retrieve and use a file either he or another researcher has previously used or created (see 3.4.0 Input-Output Module).

1.4.0 Modes of Operation

In addition to the flexibility of modularity, input formats, and file handling, Version I of the L.A.P. will take advantage of the best features of two basic kinds of operation.

An interactive, "conversational" processing environment provides the user high flexibility with little training. He interacts with the computer by answering questions, providing the program with information about options he intends to implement for a particular run. On the other hand, a non-interactive, "batch" processing environment is significantly less expensive, because it can take advantage of "slack time" on the computer, and doesn't require costly telephone connect time. For example, one Los Angeles service bureau[3] priced interactive time at $360 per CPU hour, while batch processing was $150 per CPU hour. Language analysis processing requires considerable CPU time, since most computers are not designed for text scanning and string manipulation, but rather for numeric processing. Version I of the L.A.P. will provide interaction to collect the parametric information required for the run from the user, and batch

[3] C & C Computing, 8939 S. Sepulveda, Los Angeles, California

processing for the remainder of the run on the data base. It will do so by having an interactive module which sets up the parameters for the batch run (see 3.1.0 Interact Module and 3.2.0 Control Module).

1.5.0 Definition of Terms

Before proceeding into a detailed description of the System Design for Version I, several major terms must be defined.

Structurally, the L.A.P. will be made up of Modules, each of which will consist of a Program, Sub-Program or Routine.

A Program will perform multi-task operations. It will be made up of a number of Sub-Programs which in turn may contain several Routines each. Programs will produce complete, finished output to the user. An example of a Program is the Retrieve Module.

A Sub-Program will perform single-task operations of a complex nature. It will be made up of several Routines, each processing a portion of the Sub-Program's task. Output from a Sub-Program may be complete and go to the user, or may serve as input to another Sub-Program within a Program. An example of a Sub-Program is the Morphological Analysis Module.

A Routine will perform single-task operations of a simple nature. It will usually be a dependent part of a Sub-Program or a Program, although occasionally it may stand alone (as in the case of Interface Modules which are single Routines). Output from a Routine will be used as input data or parameters for other Routines or Sub-Programs. It will provide no output to the user directly. An example of a Routine is the Translate Module.

There will be two basic types of Modules used in Version I of the L.A.P., Function Modules and Interface Modules.

A Function Module will be either a Program, Sub-Program, or Routine which performs an often complex operation. Since each Function Module will be "un-pluggable", it is important to think of the Module in terms of the function itself rather than the way in which it is accomplished. Otherwise, it will be difficult to establish a sufficiently generalized Interface Module allowing any other Program, Sub-Program or Routine to be plugged in, as long as it performs the same function.

An Interface Module will be a Routine linking the Control Module with a particular Function Module. While the Function Module is designed to be readily "un-pluggable", the Interface Module is designed to be a more permanent part of the package skeleton. It will not be un-plugged and replaced each time its associated Function Module is replaced, but only if the conceptualization of the function itself changes.

1.6.0 Documentation Concepts

Documentation of the L.A.P. will be, like the package itself, modular. If a function module of the package is changed, because a better program has been obtained for that function, the documentation can be un-plugged along with the software and as easily replaced.

There will be a series of documents reflecting the growth of the package as a whole, and of each of its component parts. There will be three basic types of documentation associated with various stages of developmental progress. They are: Design Documentation, User Documentation, and Product Documentation.

Design Documenation will con[illegible]e L.A.P. System Design Document for each version of the package, together with the Module Design Document for each module associated with that version. Design documentation will be produced before the actual implementation of the package or any specific module. The System Design Document will contain a systematic conceptual overview of the capabilities of a particular version of the L.A.P., together with generalized descriptions of those modules to be incorporated in that version. There will be a new System Design Document for each successive update of the package as a whole. The intended audience for the System Design Document will be the general researcher who can find ways of making use of the package. The Module Design Document, on the other hand, will be of a more technical nature and will be aimed primarily at the programming staff whose responsibility it is to implement the design in a workable form.

User Documentation will consist of three basic types of documents: Module Development Announcements, User's Manual Materials, and Training Materials. Such documentation will be produced after a particular module is in operating condition and available to researchers for their use. As each function module is brought into working order, a Module Development Announcement will be released, indicating the functional capabilities of

the module, how it might be used in educational research, what kind of input it requires, and what kind of output it produces. At the same time, a User's Manual and Training Materials will be issued which will give a researcher the detailed information he needs to actually prepare data and submit a request for processing by the module.

Product Documentation will consist of detailed technical documentation, including flowcharts and actual program code for each of the modules. Such documentation will be released after all other documentation relating to the module, and will be intended for a technical audience only.

All three basic types of documentation will follow the same general format and be tagged with an appropriate numeric identifier which relates back to the original System Design Document for the version of which it is a part (see Documentation Outline, p. i). Below is an outline which indicates the general form of one such piece of documentation. It is an outline of a Module Design Document associated with Version I.

TITLE: DESIGN DOCUMENT: SCAN MODULE - L.A.P. VERSION I

ABSTRACT

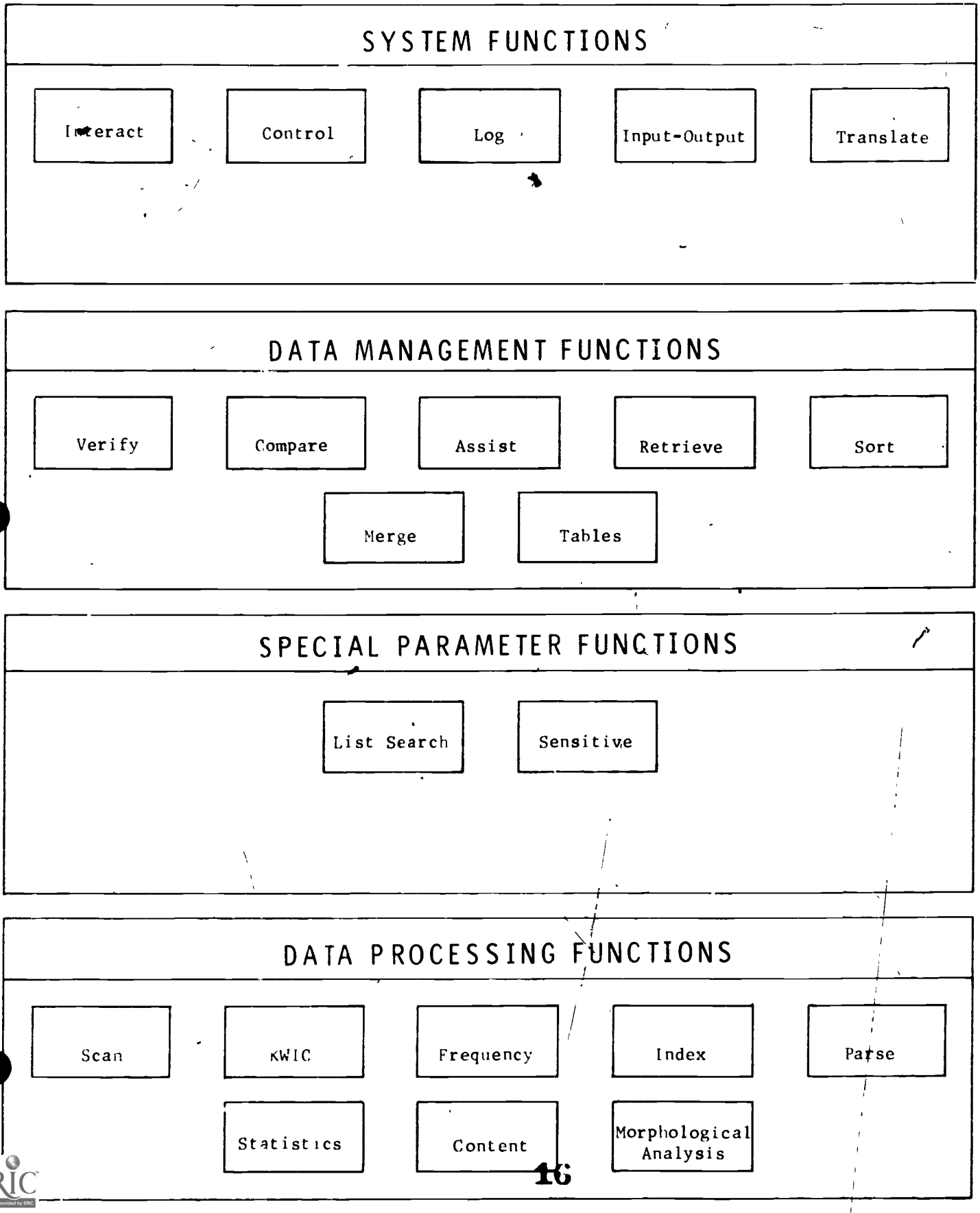
This is one of a series of technical design specifications for individual modules in the Language Analysis Package (L.A.P.).

---

1. Program Objective
2. Constraints and Limitations
3. Options and Defaults
4. Data File Specifications
   - Input
   - Output
5. Significant Algorithms
6. Significant Variables (Arrays, etc.)
7. Error and Other Messages
8. Called by and/or Calls

FIGURE 1

2.0.0 OVERVIEW OF L.A.P. DESIGN



2.1.0 Package Software Functions (see Figure 1)

The Language Analysis Package will provide four basic functions. They will be: System Functions, Data Management Functions, Special Parameter Functions, and Data Processing Functions.

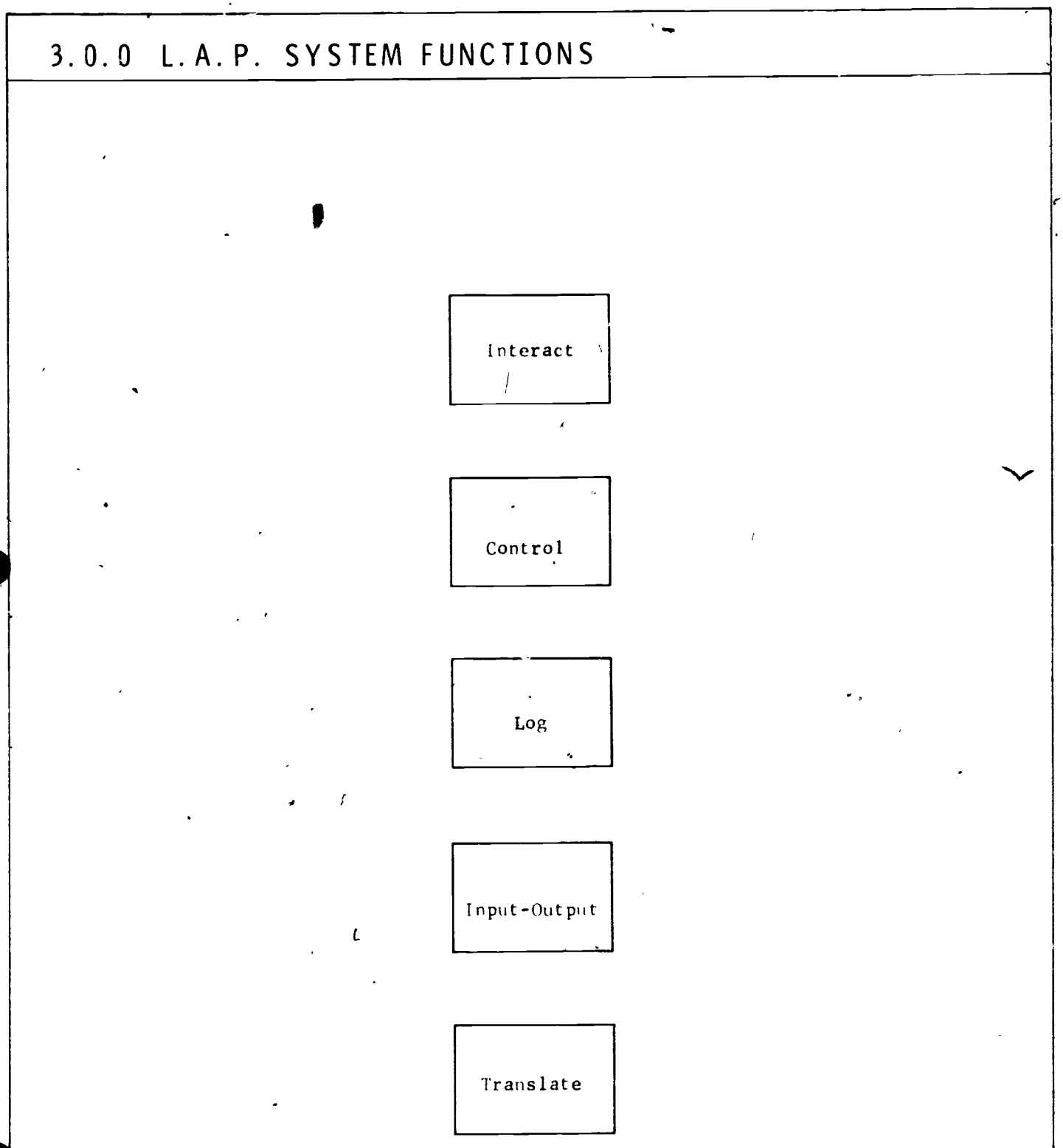
The System Functions will be concerned with the internal functioning of the package itself and the interface of the package with the user (see 3.0.0).

The Data Management Functions will be concerned with assisting the user in the preparation of his input data, data base control, and arranging his output in a form that will best facilitate his research (see 4.0.0).

The Special Parameter Functions will be primarily concerned with the definition of limitations on portions of the data to be processed, or unusual applications of package processing modules (see 5.0.0).

The Data Processing Functions will be concerned with the actual analysis of the input data, and the production of information that will further the user's research effort (see 6.0.0).

FIGURE 2


3.0.0 L.A.P. SYSTEM FUNCTIONS
Interact
Control
Log
Input-Output
Translate

3.0.0 OVERVIEW OF SYSTEM FUNCTIONS (see Figure 2)

System Functions will be those functions performed by the package which are primarily concerned with the internal functioning of the package itself and the interface of the package with the user. Unlike the Data Processing Functions, the System Functions will be written specially for the L.A.P. rather than obtained from programmers in the field of language processing. They will be tailored to the user community served by the package and be developed in a manner allowing easy modification to conform to the specific needs of new user requirements.

In addition, a high level of flexibility will be built into each of the System Functions so that a systems analyst may constantly and easily update the manner in which communication with the user takes place. In this way, the System Functions will be highly responsive to the modes of interaction most comfortable to the user community the package is serving at any given time.

For Version I, five modules will perform System Functions. They are: Interact Module, Control Module, Log Module, Input-Output Module, and Translate Module.

The Interact Module will help the user to set the parameters for his particular use of the package, and to set up the required Transaction Language Control necessary to obtain the output he needs.

The Control Module will function internally to take the Transaction Language Control and establish a decision table for use by the package in setting up a priority queue for accessing modules and routines for the present run.

The Log Module will provide the user with a record of his run, including such information as the control parameters used, and the action taken by the package based upon these parameters, as well as the date, costs, etc. of the run. In addition it will give the systems analyst information to assist in further optimizing the package as a whole.

The Input-Output Module will handle specifications made by the user concerning input and output files and provides him with a record of I-O operations.

The Translate Module will utilize information provided by the user concerning the form of his input data, and change the input constraints required of the processing module to a form compatible with the user's data. It will provide him with a record of such translations, and/or error messages if his input specifications are incompatible with the processing he has requested.

3.1.0 Interact Module

The function of the Interact Module will be to act as an interface between the user and the package. Interact will run as a front-end portion of the total package, and prepare user control parameters to be appended to the input data which will then be run in batch mode at the main facility.

It will ask the user questions about the parameters of the run he is initiating and will utilize his answers to prepare computer compatible control parameters to be read by the Control Module.

Since it is a separately functioning entity, it will serve as a training program for initiating researchers into the use of the package. As each control parameter is compiled through the question and answer process, a correctly formatted Transaction Language Control statement will be printed out. The control statements also will be passed directly to the Control Module. The control statements will be output in a form appropriate for use as input to the main package programs, such as magnetic tape.

After the researcher has used the Interact Module for a while he may find that he is sufficiently familiar with the requirements of the Transaction Language Control to prepare his own control statements without computer assistance. Certainly, such user expertise is one of the goals of the Interact Module. As a teaching, as well as a functional program, it will keep a running count of user success and failure in the question answering process, and output such information to a systems analyst when polled. The systems analyst will use such information to modify and upgrade the package for maximum success within the environment of the actual researchers making use of the system.

3.2.0 Control Module

The function of the Control Module will be to read a set of control statements containing run parameters, and to perform a decision making function for that run.

In any particular case, the user will specify which package functions he wishes to use, as well as his particular output specifications. For example, he might wish to produce a KWIC, a rank-ordered frequency count and a parsing of his text. He will indicate his needs by means of Transaction Language Control statements which preceed his input data. These TLC statements will be produced either by use of the Interact Module, or (in the case of a sophisticated user) directly. The Control Module will read the statements and compile a decision table which can be used by the program during execution to determine which Function Modules will be called in which order for that run.

If the "sophisticated" user has made syntax errors in his preparation of the statements, the Control Module will print error messages which will help him correct his errors before re-submitting the job. The Transaction Language will be designed so that typical errors, such as the omission of a comma, will be automatically corrected, allowing execution to proceed. For such corrections, a message will be printed on the output indicating the assumptions made by the Control Module, helping the user to verify that the Control Module has not misunderstood his intent. The user will have the option to specify that he does not wish execution to proceed if assumptions were necessary.

In Version I, the user will specify the form of his input data, such as record length, order and location of variables (for Statistics Module) and estimated size of data base being used. In later Versions, the Control Module will scan this information and set parameters for

the run to optimize usage of computer equipment and peripherals. Such parameters will control selection of I-O procedures, storage and access procedures, etc. Messages will accompany the output, indicating the options used in a particular run. Provision will be made for user override of the defaults.

The user will also indicate special conventions used in his data, such as a slash preceding a letter to indicate upper case, or an "@PP" preceding a character stream to indicate the beginning of a new paragraph. The Control Module will evaluate the form of the input data, and decide if sufficient information is present to allow the requested function modules to execute. If execution is possible, it will determine whether the Translate Module needs to be called. Messages will be output to the user indicating missing information which prevents execution. As in other cases, translation parameters for the particular run will accompany output. If the Translate Module is needed, the Control Module will provide the input convention information contained in the TLC statements. During execution, the Control Module will use the decision table to access appropriate Interface Modules in an appropriate order.

Any or all of the Interface Modules can be called upon by the Control Module, and the order and structure of the calling procedure need bear no resemblance to the linear thinking of the user but can be structured in terms of machine efficiency for combination and ordering of functions required by the particular run.

3.3.0 Log Module

The function of the Log Module will be to provide the user with a summary of the processing in the present computer run. It will provide the systems analyst with information concerning those subsections of the package getting the heaviest usage, allowing appropriate optimizations.

The following information will be included on a Version I user log sheet:

- The user's identification (name, cost center, etc.)
- The date and time
- A listing of the TLC statements used
- A listing of modules and routines called upon
- Execution time
- Input and output devices utilized

The following information will be included on a Version I systems analyst log sheet:

- All the items found on a user log sheet.
- Breakdown of in and out times for each procedure
- Size of data base
- Later versions will also show such things as internal storage allocations used

3.4.0 Input-Output Module

The function of the Input-Output Module will be to allow the user to specify which devices will handle his input and output, and to provide the Log Module with a record of the I-O operations. It will

work in conjunction with the Control Module, as a Sub-Program. Default input will be from punch cards, and default output will be to high speed line printer. Input may also come from magnetic tape, or disk. The Input Module will also be used when input for one module consists of output from another (such as the Retrieve Module).

All output will normally go to the printer; however, users may have particular needs or preferences and may elect to use another output device. For example, the user may wish to save his output in some computer compatible form for later input to some other program. If so, he will specify output to magnetic tape, or punched cards.

Another, although complex use of the output module, will occur when the user wants his output to serve as input for another module within the package itself. Here, the output module not only will put the output onto a selected device, but also will put the data into an appropriate storage location within the system.

3.5.0 Translate Module

The function of the Translate Module will be to provide the interface between the user's data and the data conventions required by the particular modules he wishes to use. Like the I/O Module, it may be viewed as a subprogram of the Control Module.

Since it is extremely costly to convert a large data base to another format, the Translate Module will work in the opposite direction, converting the relatively few program conventions to the format in which the data exists. Such a conversion will be accomplished in the following manner:

Each of the modules will have an array associated with it which is accessible to the Translate Module. The arrays will each have a dimension of 256, corresponding to the 256 possible 8-bit codes. Each position in the array will hold an octal number equivalent to the new value (the data dependent value) which should be utilized by the particular Function Module for the current run. The Translate Module will set up the arrays for those Function Modules being called by the current run, using the old value (the Function Module dependent value) as a subscript to locate the appropriate position within the array into which to store the data dependent value. For example, if the KWIC_Module is written to expect a slash ("/") preceding each character which is to be taken as upper case, and the user's data has been prepared with a dollar sign serving the same function, an octal 133 (equivalent to an EBCDIC "$") will be placed in position 97 of the array associated with the KWIC program, since 97 is the EBCDIC decimal equivalent of a slash ("/").[3]

Each of the modules will have a specially prepared subroutine that initializes each of the program dependent variables (such as a variable "capital") to the value contained in the appropriate position in the array associated with that module.

---

[3] EBCDIC codes have been used, since the UCLA computer facility is a likely one on which to set up the package. The same algorithm could be used with a computer which uses ASCII, with an octal 040 being placed in position 47 of the array.

The utilization of the general purpose array will allow great flexibility when a new Function Module is to be added or substituted in the system, since a completely different set of input requirements may be accomodated without modification of other modules in the package.

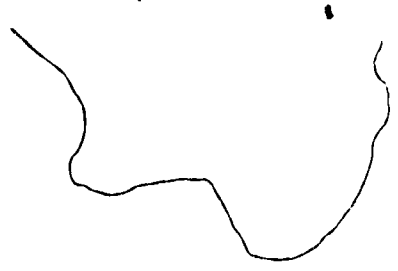
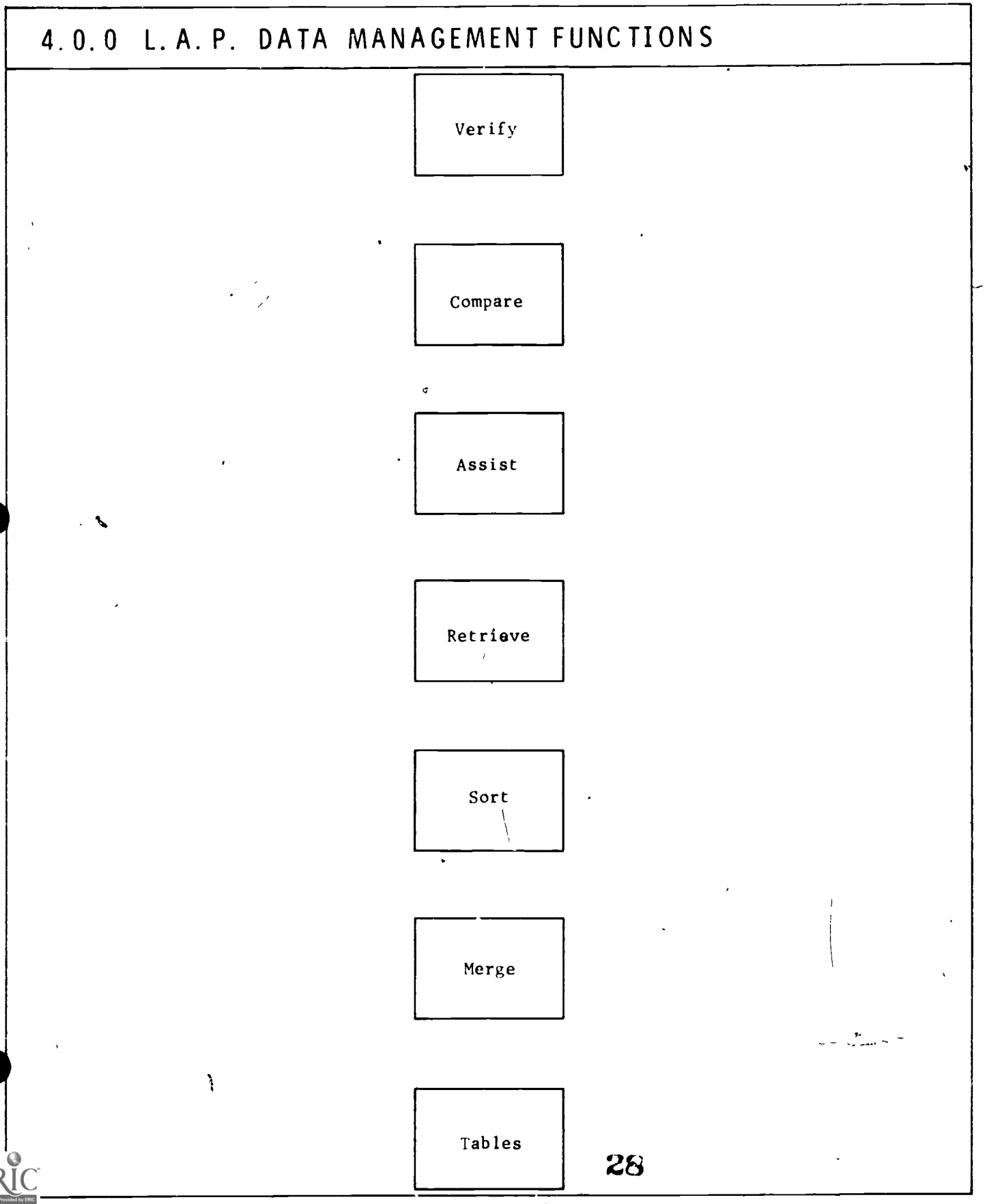


FIGURE 3


4.0.0 L.A.P. DATA MANAGEMENT FUNCTIONS
Verify
Compare
Assist
Retrieve
Sort
Merge
Tables

4.0.0 DATA MANAGEMENT FUNCTIONS (see Figure 3)

Data Management Functions are those functions performed by the package which will be primarily concerned with assisting the user in the preparation of his input data, and arranging his output in a form which will best facilitate his research. Seven modules will perform Data Management Functions. They will be Verify Module, Compare Module, Assist Module, Retrieve Module, Sort Module, Merge Module, and Tables Module.

The Verify Module will provide the user with tests of keypunching accuracy, and will flag obvious errors such as the use of "L" instead of "1" within a numeric string.

The Compare Module will test one text against another and will flag differences which occur.

The Assist Module will allow the user to have the computer identify and flag, in his input, specific items on which he needs to take special action by hand coding.

The Retrieve Module will give the user the capability of pulling out a smaller section of a large data base according to specified criteria which apply to that section and not to other sections of the total data base. Such a retrieved subsection can be used as input to other modules of the package.

The Sort Module will rearrange the output from a processing module into an order defined by the user.

The Merge Module will rearrange the output from various previous processings into a single output in a form defined by the user.

The Tables Module will prepare output in tabular form, according to row and column specifications given by the user.

4.1.0 Verify Module

The function of the Verify Module will be to check on keypunching accuracy where errors might invalidate research results. The user will be able to specify in a TLC option statement one or more of the following data verifications:

- Numerics imbedded in alphabetic strings.
- Alphabetics imbedded in numeric strings.
- Unmatched parentheses or other paired delimiters (user must specify what delimiters).
- Strings greater than 16 characters in length.
- Use of "illegal" characters (user specified).

Default will be for the program to check all the items. Output from the module will be in the form of location information and type of error, with enough context given to clearly identify the error for correction.

4.2.0 Compare Module

The function of the Compare Module will be to allow the user to obtain information on the correspondences between two texts. Output will be in the form of an alphabetical list of words appearing in text one with their locations, followed by a list of words in text two which appear at the same relative locations, but are different from those appearing in text one.

Such lists will be headed with the user supplied identifier for each of the texts compared, and the date the comparison was made. An overall consistency quotient will be given, which is the ratio of identical words/total words.

4.3.0 Assist Module

The function of the Assist Module will be to allow the user to have the computer identify and flag in his input specific items on which he wishes to take special action by inserting hand coded information.

The user will specify alphabetic or numeric strings which he wishes to have identified and flagged, and he will identify one of two ways in which he wishes them flagged for his special handling (either by the insertion of 10 blanks immediately following the item, or by the insertion of a user defined special flag symbol).

The Assist Module will scan the text for the specified strings and produce a new text file with flags included. By use of the Input-Output Module, the user can specify the device on which the file is to be saved. Default is to print the file on the line printer. The user may specify the number of copies of output he wishes.

4.4.0 Retrieve Module

The function of the Retrieve Module will be to allow the user to search a sequential or an inverted file using either a Boolean combination of terms, or a list of numeric identifiers and to obtain subsets of information contained within that data base.

A sequential file is one arranged in "normal" order, with a series of sets of related information following each other sequentially. An example of such a file would be a personnel data base that would contain a sequence of sets of information such as name, address, education, salary, date employed, etc. for each employee.

An inverted file is similar to an index in many ways. It is arranged by sub-category, and lists the location of all occurrences of information relating to the sub-category. For example, such a file would have an entry for salary = $15,000 and would list the locations of full references on all employees with that yearly income.

A search utilizing a Boolean combination of terms could yield such information as all those references where AGE = 30 and EDUCATION = MA OR PHD AND SALARY = $12,000.

4.5.0 Sort Module

The function of the Sort Module will be to allow the user to specify particular regular orderings for his input or output.

Version I options will include:

- Ascending (i.e. A-Z or 1-1,000).
- Descending (i.e. Z-A or 1,000-1).
- Combination (i.e. several sort fields, with different options for each.

Later versions will include an option for reverse word (ex. all words ending in d together, with ed words before id words, etc.)

The user will specify the fields on which he wishes sorts to be performed, with the default being one field only consisting of the first 16 characters of the record. He will specify the type of sort he wishes. Certain modules which regularly require sorted output will have their own defaults. For example, KWIC output will be sorted in ascending alphabetical order on keywords, and ascending numeric order on locations if keywords are the same; frequencies will be sorted in ascending alphabetical order on the word field if the "Alpha" option is invoked, and descending numeric order on the frequency field if the "Rank" option is invoked.

4.6.0 Merge Module

The function of the Merge Module will be to rearrange the output from various previous processing steps into a single output. The most common use of this module will be for the production of merged KWIC and merged Tables, although it may also be used with such modules as the List Search Module to help the user establish larger dictionaries.

4.7.0 Tables Module

The function of the Tables Module will be to allow the user to reformat information gathered from the Data Processing Modules, in order to make relationships within the output more easily apparent.

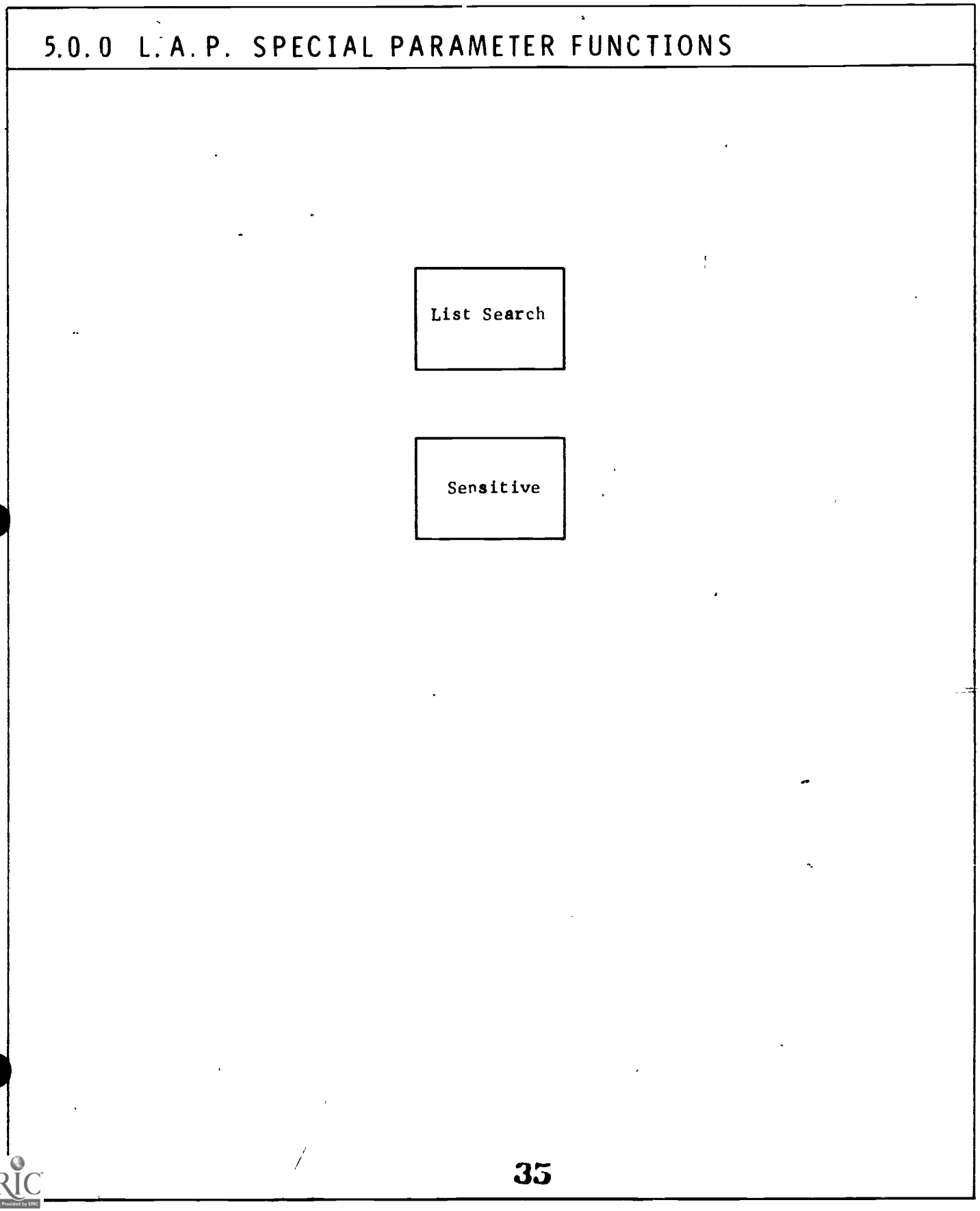
For example, if the user has obtained frequencies and percentages of numeric codes by using the Frequency Module, he will be able to make a table where closely related codes appear in vertical rows and the sub-categories of those codes appear in horizontal columns.

In the example below, codes 10-19 appear in row 1, 20-29 in row 2, and 30-39 in row 3. Relationships between the number of occurrences and percentages of codes 15 and 25 may become clearer by looking at column 5 (subcode 5).

	SUBCODE 0		SUBCODE 1		SUBCODE 2		SUBCODE 3		SUBCODE 4		SUBCODE 5		SUBCODE 6		SUBCODE 7		SUBCODE 8		SUBCODE 9	
	OCC	%	OCC	%	OCC	%	OCC	%	OCC	%	OCC	%	OCC	%	OCC	%	OCC	%	OCC	%
CODE 10																				
CODE 20																				
CODE 30																				

Version I of Tables Module will allow the user to indicate appropriate headings for his tables, and to specify which values shall appear in row and column positions.

FIGURE 4


5.0.0 L.A.P. SPECIAL PARAMETER FUNCTIONS
List Search
Sensitive

5.0.0 SPECIAL PARAMETER FUNCTIONS (see Figure 4)

Special Parameter Functions will be those functions performed by the package which are primarily concerned with definition of limitations on portions of the data to be processed, or unusual applications of package processing modules. In Version I two modules will perform Special Parameter functions. They are: List Search Module and Sensitive Module.

The List Search Module will allow the user to define lists of words or phrases which shall be either included or excluded from the processing requested.

The Sensitive Module allows the user to specify particular contexts within which, and only within which, processing is to be done.

5.1.0 List Search Module

The function of the List Search Module will be to allow the user to specify at run time particular strings (alphabetic or numeric) which shall be either included or excluded from the processing. Such sets of strings will be input in the form of a list which preferably will be alphabetized, and which will be preceded by a system control card specifying INCL or EXCL.

INCL. Specifies that processing shall be done only on strings which appear in the following list. All other strings shall be ignored.

EXCL. Specifies that processing shall be done on all strings except those which appear in the following list. Those strings in the list shall be ignored during processing.

The INCL - EXCL option will allow the user to optimize production costs by processing only the data which is essential to his particular application. The INCL - EXCL option should be used with care, especially in conjunction with Sensitive, Content and Parse since errors can occur if data pertinent to these processors is omitted through excessive limitation of the data to be included as input.

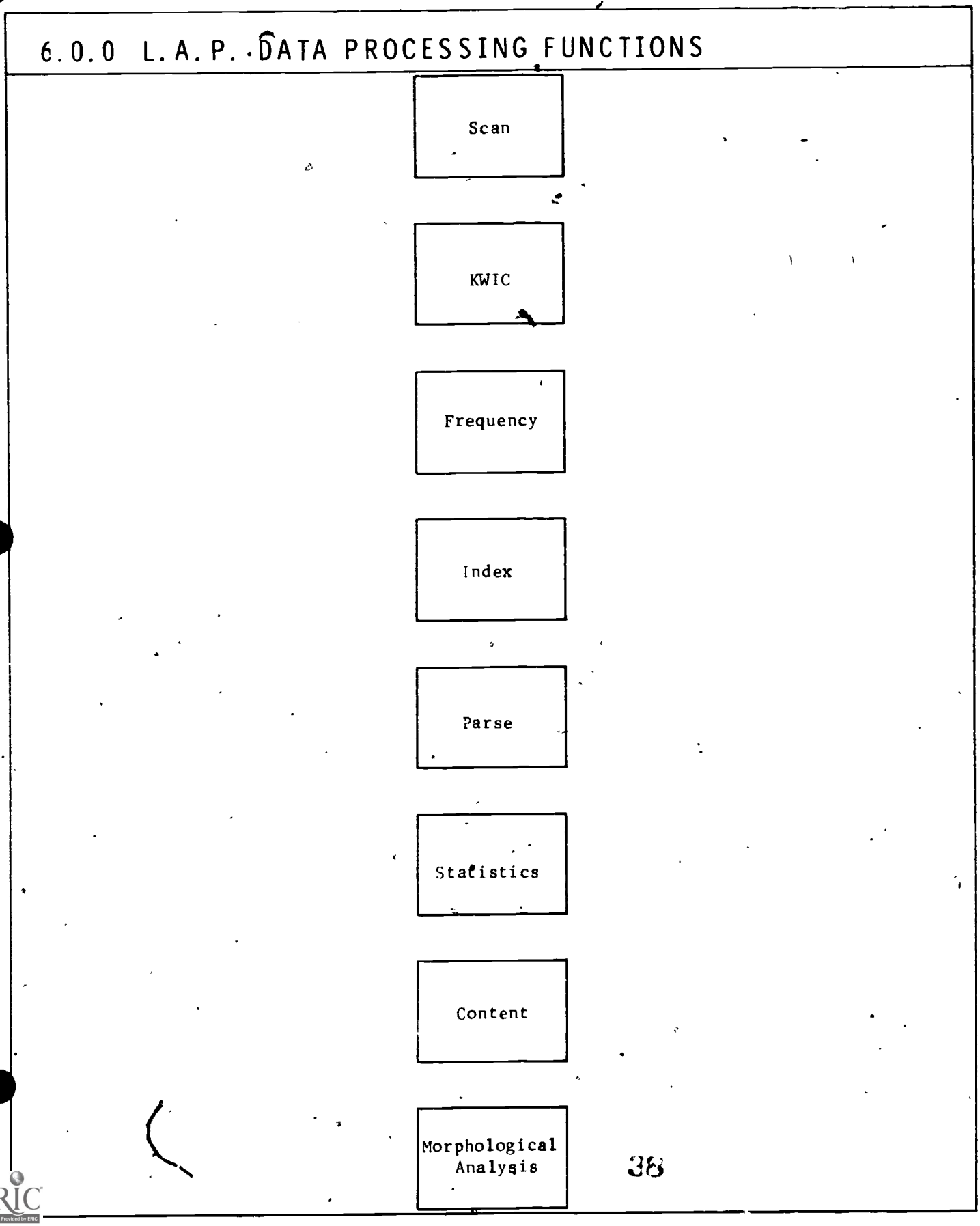
5.2.0 Sensitive Module

The function of the Sensitive Module will be to allow the user to define adjacent contexts which specify parts of the input text to be processed. For example, a user might want to process the word "reading" if and only if the word "remedial" preceded it. Or he might wish to process only the words occurring within the Title of a library citation.

The user will specify:

- The string (alphabetic or numeric) which provides the Key for processing.
- The string (alphabetic or numeric) which provides the sensitive context.
- The maximum distance (in number of characters or number of words) from the Key within which the sensitive context must occur.
- The direction (left or right) from the Key in which the sensitive context must lie.

FIGURE 5


6.0.0 L.A.P. DATA PROCESSING FUNCTIONS
Scan
KWIC
Frequency
Index
Parse
Statistics
Content
Morphological Analysis

6.0.0 DATA PROCESSING FUNCTIONS (see Figure 5))

Data Processing Functions will be those functions performed by the package which will be primarily concerned with the actual analysis of the input data, and the production of information which will further the user's research effort. Eight basic modules will perform Data Processing functions. They are: Scan Module, KWIC Module, Frequency Module, Index Module, Parse Module, Statistics Module, Content Module and Morphological Analysis Module.

The Scan Module will be the heart of the data processing functions. It will read text in, scan it character by character, and establish word boundaries, etc.

The KWIC Module will produce a Keyword in Context (Concordance) listing of the data, allowing the user to specify any one of two basic output formats, and the desired length for the before and after contexts.

The Frequency Module will produce information on the number of occurrences and percentages for words, phrases, and codes within the text. The user may specify alphabetical ordering or rank ordering.

The Index Module will produce an alphabetical index of the locations of references to particular words, phrases or codes. The user will be able to specify the terms in which locations will be given.

The Parse Module will provide the user with information concerning the grammatical structure of his input text, in the form of a tree arrangement of surface and deep structures.

The Statistics Module will produce numerical analyses of such items as means and standard deviation of word length, sentence length, paragraph length and numerous others.

The Content Module will allow the user to develop semantic categories, and test his data against them.

The Morphological Analysis Module will allow the user to obtain listings of root words contained within his text. It will be used primarily in conjunction with other processing modules, such as Parse.

6.1.0 Scan Module

Version I of the Scan Module will function as a "front end" for several of the processing modules (KWIC, FREQ, etc.). Since it will be heavily used, and since it performs its decision making function by scanning character by character through the input text, special attention will be paid to problems of optimization. Its major objective is threefold. It will read in data as necessary from the input stream; it will find and return to the calling program the beginning and end points of a word; and it will determine if the delimiter(s) following a word is a "special" character which signals the necessity of some kind of special handling involving an additional subroutine call.

Depending on the calling module, Scan may return either the beginning and ending points of the word, or the word itself for storage in an array.

The user will be able to specify a definition of characters to be considered as valid word-parts. Default will be to have the alphabet, apostrophe, hyphen and numbers considered as word-parts with all other characters considered as delimiters. In addition, the user will be able to specify which of the delimiters should be considered as "special" characters.

6.2.0 KWIC Module

The function of the KWIC Module is to produce Keyword-in-Context (Concordance) listings from the input text. Location information will be broken down into four main categories: level 1 (ex. Document ID), level 2 (ex. Page), level 3 (ex. paragraph), and level 4 (ex. line). The KWIC Module will be run frequently in conjunction with the List Search Module. One such application will be when producing KWIC's on library citations where only words in the title will be processed.

The user may specify the following:

- Length of before context to be specified either in number of words or number of characters. (default is 48 characters)
- Length of keyword. (default is 16)
- Length of after context to be specified either in number of words or in number of characters. (default is 48 characters)
- Sort fields and ordering (default is keyword, location, in ascending order)
- Either of the following two output formats:
    A) one line, keyword centered
    B) two line, keyword right justified

  (default is A)

6.3.0 Frequency Module

The function of the Frequency Module will be to produce alphabetical and rank ordered computations of the number of times words and/or numeric codes occur, and what percentage of the total text is represented by each.

The user may specify the following:

- Alphabetical ordering of the list. (words beginning with "A" first)
- Rank ordering of the list. (most frequent first)
- Occurrences only.
- Percentages only.

Default will be for the Module to produce all four. The Frequency Module will be run often in conjunction with the Tables Module to produce a tabular presentation of the information in a user specified format.

6.4.0 Index Module

The function of the Index Module will be to allow the user to compile an index of locations of specific words and phrases within his text. The module will generally be used in conjunction with the List Search Module.

The index produced will be alphabetically sorted. The default for locations will be the same as for the KWIC Module (see 6.2.0). If no inclusion or exclusion list is provided by the user, a standard exclusion list will be used which consists of high frequency words such as "and", "the", "in", "by" etc.

6.5.0 Parse Module

The function of the Parse Module will be to provide the user with information concerning the grammatical structure of sentences contained within the input text.

The user may provide his own grammar and dictionary in a format compatible with the program parameters.

Output is in the form of tree-structures representing the surface and deep structures of the sentences.

The Parse Module will be run in conjunction with the Morphological Analysis and List Search Modules.

The Parse Module should be used with care, since processing costs for large input texts are often high.

6.6.0 Statistics Module

The function of the Statistics Module will be to allow the user to obtain summary information of a statistical nature concerning his input text.

Following are some of the more obvious statistical measures of text:

- Total number of occurrences of each mark of punctuation.
- Total number of words in text.
- Total number of different words.
- Ratio of unique/total words (type/token ratio)
- Frequency of words of length n (where n ranges from 1 to 24 characters).
- Frequency of sentences of length m (where m ranges from 1 to 40 words).
- Total number of sentences in text.
- Total number of paragraphs in text.
- Mean word length in sentences.
- Mean sentence length in words.

- Mean paragraph length in sentences.
- Standard deviation of word length.
- Standard deviation of sentence length.
- Standard deviation of paragraph length.
- Third moment of word length.
- Fourth moment of word length.

6.7.0 Content Module

The function of the Content Module will be to allow the user to set up dictionaries of phrases in numerous semantic categories, and to match his particular text against one or more of these dictionaries and have a category item analysis performed. A tabulation will be given by category of the matching items and the number of occurrences. For example, a user studying the relationships of Mexican American oriented state approved text books may want to make phrase dictionaries for semantic categories such as "nuclear family", "extended family", "individualism", "socialization". "present orientation", "future orientation", "patriarchy", "matriarchy", "cooperation", and "competition". By using the Content Module, he could determine the concept weighting which occurs.

6.8.0 Morphological Analysis Module

The function of the Morphological Analysis Module will be to allow the user to obtain information concerning the root words contained within his input text. Generally, this module will be run in conjunction with other modules such as Parse and List Search but the user may obtain a root word list independently of other module uses.

7.0.0 TRANSACTION LANGUAGE CONTROL (T.L.C.)

The function of the Transaction Language will be to provide the user with a method of communicating parameters for operating the computer run. It will allow the user to specify the form of his input data, the package functions he wishes to use, specific options he requires, and the form of output he wishes to obtain.

7.1.0 General Concepts

The basic design of the TLC emphasizes ease of use. There are few syntax constraints. With the exception of the initial identifying "!" in column 1, parameters may be in a relatively free format form. After the "Control Word" (RUN, FUNC, OPT, etc.), required sub-items of information may be in any order. Imbedded blanks are ignored so the user may format his TLC cards as he wishes. If a command is misspelled, the system will make a "best guess" based on the context and the first two letters, and will print its "guess" or a message indicating that it has insufficient information to continue processing.

7.2.0 Required System Commands

All TLC cards will begin with a "!" in column 1. Continuation cards will be indicated by "!" in columns 1 and 2.

The following six "Control Words" will be required for any run made with the package. The Control Word must be the first item following the exclamation point, but need not be in any particular column.

- !RUN
- !INPUT
- !FUNC
- !OPT
- !OUTPUT
- !DONE

!RUN

The !RUN card will tell the system that a run is being initiated. Any information punched on the !RUN card from columns 6-80 or on continuation cards will be used as run identification and will be printed at the top of each page of output. No more than two continuation cards will be allowed.

!INPUT

The !INPUT card will specify any special parameters of the user's input. The following information must be included on the !INPUT card and/or its continuations:

- Input record length
- Input device (magnetic tape, cards, etc.)
- Code used (ASCII, BCD, EBCDIC)
- Special conventions

  ('PG = NEW PAGE; '/' = UPPER CASE; etc.)

Any number of continuation cards may be used.

!FUNC

The !FUNC card will indicate which Data Management and Data Processing functions the user wishes to employ. Following is a list of acceptable functions which may be specified by the !FUNC card:

- VERIFY
- COMPARE
- ASSIST
- RETRIEVE
- SORT
- MERGE
- KWIC
- FREQUENCY
- INDEX
- PARSE
- STATISTICS
- CONTENT
- MORPH ANAL
- LIST SEARCH

One !FUNC card will be required for each function employed, and it must be followed immediately by its associated !OPT card. However, the two-card "sets" (!FUNC and !OPT) may occur in any order.

!OPT

The !OPT card immediately follows the !FUNC card and is associated with it. An !OPT card without a preceding !FUNC card will result in an error message and vice-versa. The !OPT card will associate a list of appropriate options with the functions requested. For example, one option of the KWIC function is the use of an inclusion or exclusion list; one option of the Frequency function is a rank-ordered list.

!OUTPUT

The !OUTPUT card will specify what output the user wishes to receive, and will allow him to direct his output to whatever device he desires. He may suppress Log output and summary diagnostics if he wishes. It is this card which will specify the destination if the output from one function (for example Frequency) is to be used as the input for another function (for example Tables).

!DONE

The !DONE card indicates that all TLC cards have been input and the user is requesting compilation and execution. All cards after the !DONE card will be considered input data.

8.0.0 SUMMARY AND CONCLUSIONS

The most important features of the Language Analysis Package will be adaptability of modular design and flexibility of user-defined options.

Individual portions of the package will be used without the user being penalized by high processing costs due to superfluous modules being called.

Development costs will be held to minimum by utilizing existing programs already written and debugged by specialists in the field of natural language processing.

Documentation also will be modular, and will develop organically with the growth of the package. At every stage documentation will be produced reflecting latest developments and most recent revisions.

The value of the L.A.P. will lie in the research opportunities it will open for the non-computer oriented researcher, who previously had no ready tool to do analysis of textual data.

APPENDIX A

Bibliography

Bordon, George A. and Watts, James J. "A Computerized Language Analysis System," Computers and the Humanities, V:3 (Jan. 71) pp. 129-142

Borko, Harold, Automated Language Processing, John Wiley, (1967)

Burelbach, Frederick M. (ed.), Proceedings: Computer Applications to Problems in the Humanities, Brockport, (1969)

Computer Studies in the Humanities and Verbal Behavior (various issues)

Computers and the Humanities (various issues)

DeBoer, Aient. A Modular Program for Reference Retrieval from Bibliographic Data Bases, unpublished M. A. Thesis, University of California at Los Angeles, Information Sciences (1970)

Dewar, Hamish (et. al.), "A Program for the Syntactic Analysis of English Sentences," in Communications of the ACM, (August, 1969), p. 476

Dixon, W. J. (et. al.) Biomedical Computer Programs, University of California Press, (1970)

Dodd, George, "Elements of Data Management Systems" in Computing Surveys, (June 1969).

Early, Jay, "An Efficient Context - Free Parsing Algorithm," in Communications of the ACM, (February, 1970) p. 94

Fisher, Gerald. The SCORTXT Program for the Analysis of Natural Language, University of Connecticut, Bureau of Educational Research

Foster, J. M., Automatic Syntactic Analysis, American Elsevier, (1970)

Frey, Regina and Gould, Laura, Tricon Concordances, Linguistic Automation Project, Yale University, (1967)

Friedman, Joyce, "Fortran Implementation of the Kay Context - Free Parser," Working Paper In Computational Linguistics, (June, 1970)

Garvin, Paul L. Natural Language and the Computer, McGraw Hill, (1963)

Gerbner, George (et. al.) The Analysis of Communication Content, John Wiley, (1969)

Glantz, Richard, "SHOEBOX - A Personal File Handling System for Textual Data," MITRE Corp. (1970)

Gross, Louis and Walker, Donald, "On-Line Computer Aids for Research in Linguistics" in Information Processing 68, North Holland Publishing Company, (1969), p. 1531

Hays, David G., Introduction to Computational Linguistics, American Elsevier, (1967)

Hays, David G., Readings in Automatic Language Processing, American Elsevier, (1966)

Holsti, Ole R., Content Analysis for the Social Sciences and Humanities, Addison-Wesley, (1969)

Horowitz, Floyd R. (ed.), Report of the Conference on Computer Technology in the Humanities, National Science Foundation GJ-505, (September 1969)

IBM Symposium on Introducing the Computer into the Humanities, IBM, (1969)

IBM System/360, Disk Operating System: Tape and Disk Sort/Merge Programmer's Reference Manual, SC28-6695

Jahoda, Gerald, Information Storage and Retrieval Systems for Individual Researchers, John Wiley, (1970)

Kline, Lanaii. Essay Word Count and Statistics Program, TN 5-71-56 (August 18, 1971)

Kucera, Henry and Francis, W. Nelson, Computational Analysis of Present-Day American English, Brown University Press, (1967)

Lamb, Sydney and Gould, Laura, Type Lists, Indexes and Concordances from Computers, Linguistic Automation Project, Yale University, (1967)

Language and Automation (various issues)

Leed, Jacob, The Computer and Literary Style, Kent State University Press, (1966)

Linguistics in Documentation (current abstracts) (various issues)

Meadow, Charles T., Man-Machine Communication, John Wiley, (1970)

Minker, Jack (ed.) Proceedings of the Symposium on Information Storage and Retrieval, University of Maryland Press, (1971)

Minsky, Marvin, Semantic Information Processing, MIT Press, (1968)

Miroff, Linda, "Three PL/1 Subroutines for Text Processing," Institute of Library Research, UCLA, (1970)

Nie, N. (et. al.) Statistical Package for the Social Sciences, McGraw-Hill, (1970)

Porch, Ann. A Hardware Configuration for a Flexible, Multi-Purpose Data-Entry Station (May 13, 1971)

Porch, Ann. People and Projects in Natural Language Processing: A Preliminary Bibliographic Directory, TM 5-71-10 (August 5, 1971) 107 pp.

Porch, Ann. A Preliminary Design for a Language Analysis Package, TN 5-71-74, (August 18, 1971), 14 pp.

Quillian, Ross, "A Teachable Language Comprehender: A Simulation Program and Theory of Language," in Communications of the ACM, (August, 1969) p. 459

Reichert, Richard and Olney, John, Two Dictionary Transcripts and Programs for Processing Them, SDC, (1969)

Robison, Harold R., "Computer Detectable Semantic Structures," in Information Storage and Retrieval, (1970), p. 273

Salton, Gerald (ed.), The SMART Retrieval System: Experiments in Automatic Document Processing, Prentice-Hall, (1971)

Sedelow, Sally (et. al.) Automated Analysis of Language Style and Structure, Chapel Hill, N.C., (1970)

Shapiro, Stuart Charles, The MIND System: A Data Structure for Semantic Information Processing, Rand Corp. (August, 1971)

Singh, Jagjit, Great Ideas in Information Theory, Language and Cybernetics, Dover, (1966)

Von Glaserfield, Ernst and Pisani, Paolo, "The Multistom Parser for Hierarchial Syntactic Structures," in Communications of the ACM, (February, 1970), p. 74

Walker, Donald E., "Computational Linguistic Techniques in an On-Line System for Textual Analysis," Information System Language Studies Number 22, (1969)

Winograd, Terry, Procedures As a Representation for Data in a Computer Program for Understanding Nacural Language, MIT Press, (1971)